Problem Solving by Computer: Project 3

Karan Kapotra

July 27, 2020

The following three questions all pertain to the same data set of a retailer and a list of orders and corresponding customers curated over a four-year period. The questions relate to understanding the customers interactions with the company and trying to understand their habits from the purchases.

1 Logistic Regression

1.1 Description

The first task is to create a logistic regression model to predict how likely the order is to be returned and the rating that the order is given by the customer. We were given the logistic regression equation shown below, however we had to calculate the coefficients α and β, which related to our problem.

$$P(r) = \frac{1}{1 + exp(-\alpha r - \beta)}$$

1.2 Methodology

First of all, we process the csv file of the orders into our program. This creates a table, akin to an excel table with headers and cells with information pertaining to the orders. The information we received was as follows: the date the order occurred, individual customer IDs, the category of the order (for instance clothing, electronics, etc), the value of the product, what rating the order was given (0 if no rating was given), and if the order was returned or not. For this task, we want only the customers which have ever returned an order, therefore we first have to find all the orders which have been returned and then the customer IDs related to the each order, thus finding uniquely all the customers which have ever returned an order. Using this list of unique customers, we can find all the orders which have been made by a customer who has completed a return. We want only the orders which have been rated therefore we can also disregard any rating which is zero.

We pass two hyperparameters into the matlab optimisation function fminsearch which we run over the logistic regression equation. One is a binary list of the orders, with a 1 if it is returned and a 0 if not. The other is the rating of the corresponding order which is made.

1.3 Algorithm

```
%Read in the table of orders
Orders = readtable('purchasing_order.csv');

%Find the returned orders only
ReturnsOnly = Orders(strcmp(Orders.Return, 'Y'), :);

%Find the unique customers who have ever returned anything
CustomerIDs = unique(ReturnsOnly.Customer_ID);

%Find only the orders of the corresponding customers and rating > 0
Orders = Orders(ismember(Orders.Customer_ID, CustomerIDs), :);
Orders = Orders(Orders.Rating > 0, :);


p = strcmp(Orders.Return, 'Y');
h = Orders.Rating;

lr_par = fminsearch(@(a)logreg(a,h,p),[0 0]);

hx = linspace(0,5,1000);

plot(h,p,'*',hx,1./(1+exp(-lr_par(1)*hx-lr_par(2))));
lg = legend('Raw Data','Logistic Regression');
axis([0 6 -0.1 1.1]);
xticks([1:5]);
yticks([0 1]);
yticklabels({'No', 'Yes'})
xlabel('Rating');
ylabel('Return');

%The logistic regression algorithm
function S = logreg(a,h,p) % a = [a , b]

S = 0;
for k = 1:length(h)
    S = S + (p(k)-1/(1+exp(-a(1)*h(k)-a(2))))^2;
end

end
```

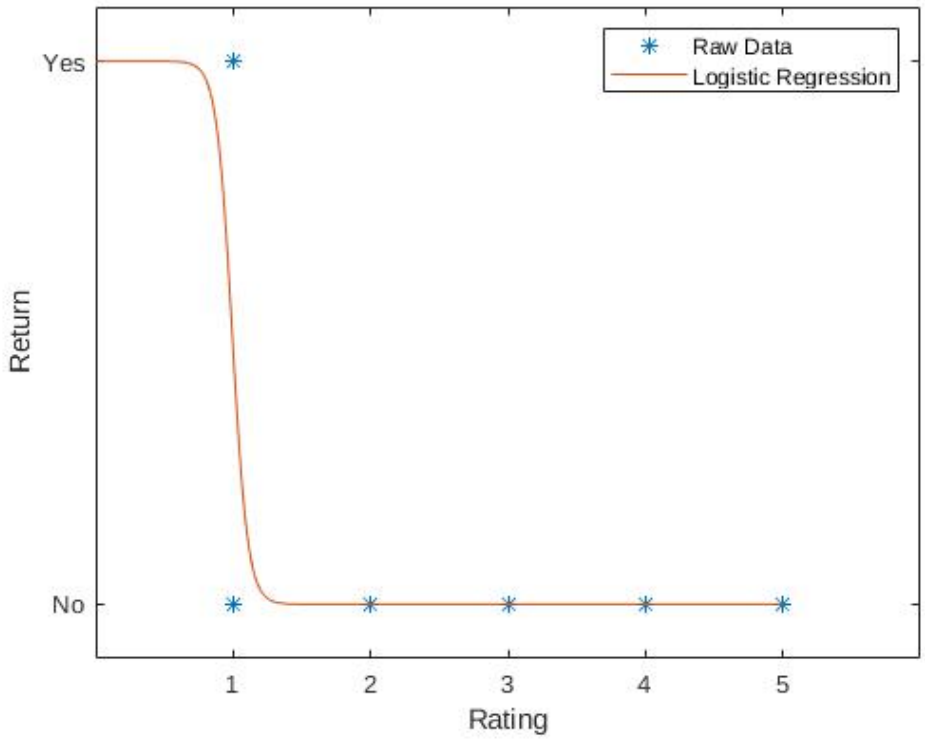
1.4 Analysis

Running the optimisation function with the orders information given and the logistic regression algorithm, we find α and β and the logistic regression equation is calculated to be as seen below.

$$P(r) = \frac{1}{1 + exp(17.54241r - 17.41880)}$$

From the use of the matlab function fminsearch, we minimise the area under the graph, which is the plot of the rating against the return, which are our hyperparameters.

The figure below shows the relationship between the rating and the probability of returning the order. It is clear to see the as the rating of the order increases, the chances of the order being returned drop significantly. Which we can assume supports any real world cases.



2 Future Purchases Based on Past Returns

2.1 Description

For this task, we have to prove or disapprove the following statement using the data set available to us. The statement is as follows: "It is believed that when a customer has returned

a product, he or she less likely to make future orders from the company". This will be quantified by finding the total values of the orders before the return divided by the total value of orders over the entire four-year period bar the returned (hence refunded) orders.

2.2 Methodology

As with the first problem, we have to firstly transform the table and extract the information we need from the orders csv. We again, find the customer IDs related to the orders which were returned. We will only be working with this subset of customers as the statement only pertains to customers which have refunded an order. Therefore, any customer which has never refunded an order is removed from our orders table. Following this, we sort our orders table by customer ID, so we have all the customers orders in line with each other and then sort by date so we know which orders were before and after the refunded order. Now that we have the orders in a format we can work with, we iterate through each customer and each order which they made.

We iterate through all the orders, checking if the order is of the current customer in the list. If not, we go to the next customer. Firstly we check if the current order is returned or not. Then we add on to our variable for total value, the price of the current order, as long as it is not a refunded order. If we have found a refunded order, we then add the price on to a variable which holds the total value of orders after the refund.

We now simply find the probabilities using the two variables and find the mean of them all.

2.3 Algorithm

```
%Read in the table of orders
Orders = readtable('purchasing_order.csv');

%Find the returned orders only
ReturnsOnly = Orders(strcmp(Orders.Return, 'Y'), :);

%Find the unique customers who have ever returned anything
CustomerIDs = unique(ReturnsOnly.Customer_ID);
CustomerIDs = sortrows(CustomerIDs);
noOfCustomers = length(CustomerIDs);

%Find only the orders of the corresponding customers
Orders = Orders(ismember(Orders.Customer_ID, CustomerIDs), :);

%Sort by CustomerID then Date
sortedOrders = sortrows(Orders,{'Customer_ID','Date'});

probabilties = zeros(noOfCustomers, 1);
x = 1;
noOfOrders = height(sortedOrders);
returnFound = 0;
```

```
totalValue = 0;
valueAfterReturn = 0;

for count = 1:noOfCustomers
    while(sortedOrders.Customer_ID(x) == CustomerIDs(count))

     %Check if we have found the returned order yet
     if (strcmp(sortedOrders.Return(x), 'Y') == 1)
         returnFound = 1;
     end

     %Total value of orders without the returned order
     if (strcmp(sortedOrders.Return(x), 'Y') == 0)
     totalValue = totalValue + sortedOrders.Product_Value(x);
     end

     %Value of orders after the returned order
     if (returnFound == 1 && strcmp(sortedOrders.Return(x), 'Y') == 0)
         valueAfterReturn = valueAfterReturn + sortedOrders.Product_Value(x);
     end

     if (x < noOfOrders)
         x = x + 1;
     end
     if (x == noOfOrders)
         break;
     end
    end

     %Caluclate the probabibility
     probabilties(count, 1) = valueAfterReturn / totalValue;

     %Reset values for the next customer
     totalValue = 0;
     valueAfterReturn = 0;
     returnFound = 0;
end

%Find the mean of the probabilities
meanReturns = mean(probabilties);
```

2.4 Analysis

The output of the program returns a mean of the outputs for all the customers. The outputs showed us, how much of the total value of the orders was made after the refund. The average which is calculated at the end of the program is 0.5041. This tells us that the average total of orders before and after the refund is nearly equal, thus disapproving the previous statement. We can say that if a person has refunded an order, it does not effect their future orders or

that there is no correlation.

3 Ranking Top 100 Customers

3.1 Description

In this task, we have a 100 coupons to give to customers of this retailer. The coupons give 30 percent off of the true value of a product in the clothing section. However, the problem is to find which 100 customers are given the coupon, taking into account expenditure and customer loyalty. Therefore, using the two criterion: average product value in category 'C' and their average rating, we need to find a weighting of both which allows us to rank the customers so we know which 100 are the most suitable to be given the coupon.

3.2 Methodology

The main calculations of this problem are the two ranking criterion and finding the ultimate ranking from the combination of the two.

Finding the two criterion is fairly straightforward, as a continuation of the previous problems. First, we remove any orders which are not of the category 'C', otherwise known as the clothing orders. Thus, we sum each customer's orders and ratings and find the mean of each. We have now calculated the two criterion individually. Now the next step is to find the combination of both to find the final ranking.

3.3 Algorithm

```
%Read in the table of orders
Orders = readtable('purchasing_order.csv');

%Find the category C only and sort the table by ID
Orders = Orders(strcmp(Orders.Product_Category, 'C'), :);
Orders = sortrows(Orders,{'Customer_ID'});

noOfOrders = height(Orders);

CustomerIDs = unique(Orders.Customer_ID);
noOfCustomers = length(CustomerIDs);

Ranking = zeros(noOfCustomers, 4);

x = 1;
totalValue = 0;
OrderCnt = 0;
AvgRating = 0;
```

```
noOfOrders = height(Orders);

for count = 1:noOfCustomers
    while(Orders.Customer_ID(x) == CustomerIDs(count))

        totalValue = totalValue + Orders.Product_Value(x);
        OrderCnt = OrderCnt + 1;

        AvgRating = AvgRating + Orders.Rating(x);


        if (x < noOfOrders)
          x = x + 1;
        end
        if (x == noOfOrders)
         break;
        end
    end

    %Track the ID, their average order value and rating
    Ranking(count, 1) = CustomerIDs(count);
    Ranking(count, 2) = totalValue / OrderCnt;
    Ranking(count, 3) = AvgRating / OrderCnt;

    totalValue = 0;
    OrderCnt = 0;
    AvgRating = 0;
end
format long

maxValue = max(Ranking(:, 2));
maxRating = max(Ranking(:, 3));

%Normalise the average orders and ratings and add them together
for count = 1: noOfCustomers
   Ranking(count, 2) = Ranking(count, 2) / maxValue;
   Ranking(count, 3) = Ranking(count, 3) / maxRating;
   Ranking(count, 4) = Ranking(count, 2) + Ranking(count, 3);
end

%Rank them from highest total to lowest
Ranking = sortrows(Ranking, 4,'descend');
```

3.4 Analysis

When approaching the final part of this task, my initial thought was to use an 'Olympic' style ranking system. This can be described as weighting gold, silver and bronze medal respectively (usually with weights of 3, 2, and 1) and multiplying by the number of each medal won and

adding them altogether to give a final score. This would be easy with our average rating and average order value, however it does not work with '0' ratings as they signify a score which was not given. Thus, I looked for another way to complete the final ranking.

We can instead normalise both the average ratings and average order values. This means finding the maximum value of the average rating and dividing each average rating by the maximum, such that each customer's average rating is now between 0 and 1. We do this similarly with the average order value. Once this is completed, we sum both normalised values for each customer and this summation is our final ranking, with the highest sum at the top. In the table below you can find the top 10 customers from this ranking and the bottom 10 of the first 100.

However this ranking method has some criticisms. If one of the average values has a extreme outlier as its maximum, this will cause a large bias in the normalisation. Also this method considers both the rating and the order value as equal weighting, however depending on the business aim, one of the rankings may be more preferred than the other. For instance, if the business wants a better marketing appearance and increase its sales through word of mouth and customer satisfaction, it may want a higher weighting on the average rating. Conversely, if they want to focus on profits, they may want to focus on the customers with the highest average order value.

Ranking	**Customer ID**	**Ranking**	**Customer ID**
1st	1014288	**91st**	1015326
2nd	1011981	**92nd**	1013142
3rd	1016309	**93rd**	1012489
4th	1014429	**94th**	1013800
5th	1012195	**95th**	1011041
6th	1016443	**96th**	1011068
7th	1015864	**97th**	1017300
8th	1014953	**98th**	1011436
9th	1014887	**99th**	1013998
10th	1011918	**100th**	1017777